Practical Data Science using R Lesson 3: Web Scraping with `XML`

Maher Harb, PhD Assistant Professor of Physics Drexel University

About the lesson

- A lot of interesting data exists in webpages
- Data in webpages is formatted to display nicely on a browser, but it may not be available for download in a convenient format (such as csv)
- In this lesson, we'll learn how to retrieve data from a webpage and organize it in a tidy format
- We'll do so using the `XML` package
- We'll also learn how to search for patterns in text using **regular expressions**

Retrieving webpages

- In Lesson 1, we used the `httr` package to download files from the web
- The same `GET` function in `httr` is used to download a webpage
- A webpage is a text file with **HTML tags** wrapped around the text
- The tags tell the browser how to format the text to display according to the intended webpage design

Billboard top 200

The HTML code of Billboard top 200

```
.
.
.
<article class="chart-row chart-row--5 js-chart-row" data-hovertracklabel="Song Hover-" data-songtitle=
<div class="chart-row__primary">
<div class="chart-row__history chart-row__history--falling"></div>
<div class="chart-row__main-display">
<div class="chart-row__rank">
<span class="chart-row__current-week">5</span>
<span class="chart-row__last-week">Last Week: 3</span>
</div>
<div class="chart-row__image" style="background-image: url(https://charts-static.billboard.com/img/1967/
</div>
<div class="chart-row__container">
<div class="chart-row__title">
<h2 class="chart-row__song">Sgt. Pepper&#039;s Lonely Hearts Club Band</h2>
<a class="chart-row__artist" href="/music/the-beatles" data-tracklabel="Artist Name">
The Beatles
</a>
</div>
</div>
```

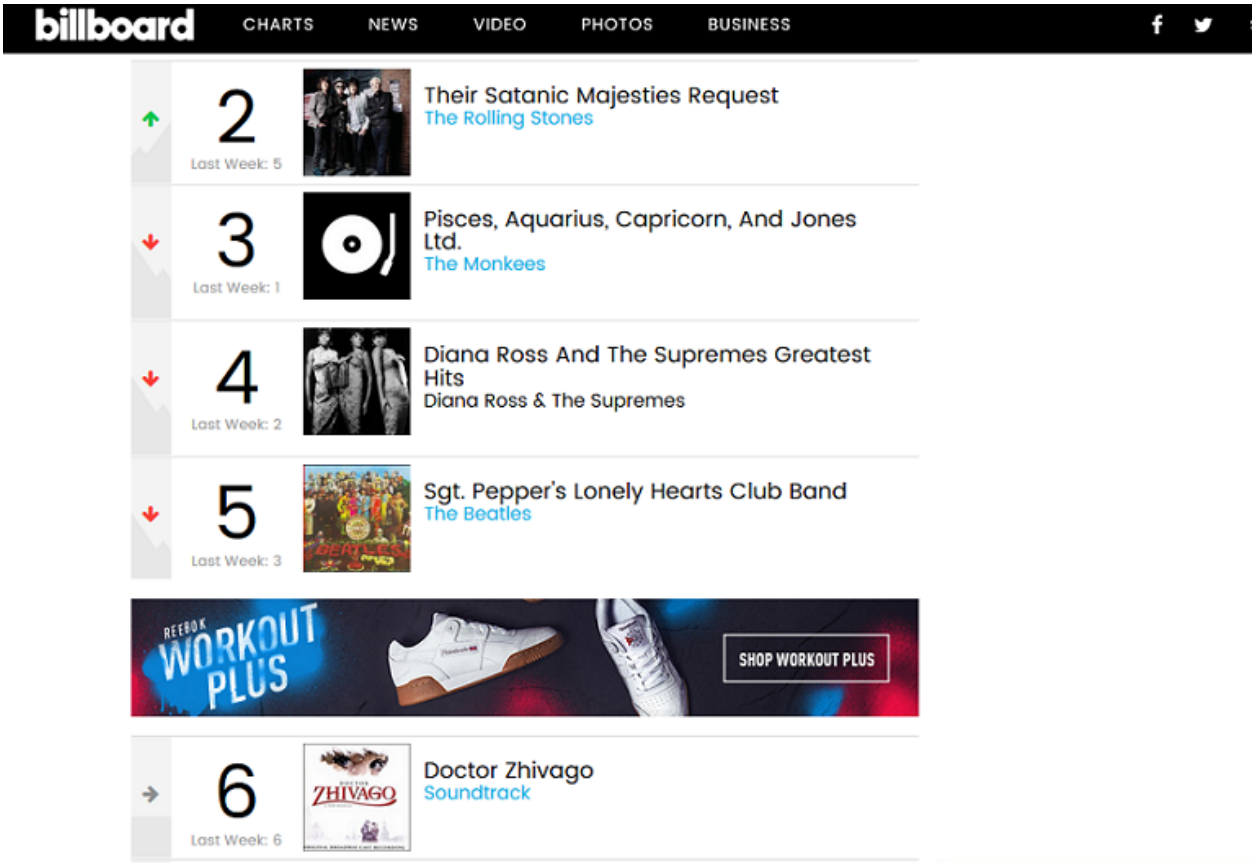

billboard
CHARTS
NEWS
VIDEO
PHOTOS
BUSINESS
2
Last Week: 5
Their Satanic Majesties Request
The Rolling Stones
3
Last Week: 1
Pisces, Aquarius, Capricorn, And Jones Ltd.
The Monkees
4
Last Week: 2
Diana Ross And The Supremes Greatest Hits
Diana Ross & The Supremes
5
Last Week: 3
Sgt. Pepper's Lonely Hearts Club Band
The Beatles
REEBOK
WORKOUT PLUS
SHOP WORKOUT PLUS
6
Last Week: 6
Doctor Zhivago
Soundtrack

Figure 1:

```
<div class="chart-row__links">
<a class="chart-row__link chart-row__link--toggle js-chart-row-toggle" href="javascript:void(0);">
<i class="chart-row__icon fa fa-angle-down"></i>
</a>
.
.
.
```

The HTML code of Billboard top 200

- The HTML code indicates that there is a good amount of structure in the tagging scheme
- Example, album titles are listed within an `<h2></h2>` tag with class name `chart-row__song`:

```
<h2 class="chart-row__song">Sgt. Pepper&amp;#039;s Lonely Hearts Club Band</h2>
```

- And album artists are listed within an `<a></a>` tag with class name `chart-row__artist`:

```
<a class="chart-row__artist" href="/music/the-beatles" data-tracklabel="Artist Name">
The Beatles
</a>
```

The HTML code of Billboard top 200

- Why do we care about the HTML structure?
- A well-structured HTML tagging scheme means that we could potentially write some script that can take advantage of the tagging structure to extract all album titles and artists
- In R, we do not need to do much parsing from scratch
- Extracting data from the HTML page is done with the `XML` package

Let's take a look at the implementation...

Extracting data from Billboard.com

Going back to the Billboard example, first we retrieve the webpage (top 200 chart on Jan 6, 1968):

```
library(httr)
library(XML)
url <- "https://www.billboard.com/charts/billboard-200/1968-01-06"
xmlpage <- htmlParse(rawToChar(GET(url)$content))
length(capture.output(xmlpage))
```

```
## [1] 10230
```

```
capture.output(xmlpage)[1:4]
```

```
## [1] "<!DOCTYPE html>"             "<html class=\"\" lang=\"\">"
## [3] "<head>"                      "<meta charset=\"utf-8\">"
```

Extracting data from Billboard.com

Next, we extract the album titles:

```
searchfor <- "//h2[@class='chart-row__song']"
album <- xpathSApply(xmlpage, searchfor, xmlValue)
length(album)
```

```
## [1] 200
```

```
head(album, 6)
```

```
## [1] "Magical Mystery Tour (Soundtrack)"
## [2] "Their Satanic Majesties Request"
## [3] "Pisces, Aquarius, Capricorn, And Jones Ltd."
## [4] "Diana Ross And The Supremes Greatest Hits"
## [5] "Sgt. Pepper's Lonely Hearts Club Band"
## [6] "Doctor Zhivago"
```

Extracting data from Billboard.com

And then we extract the artists:

```
searchfor <- "//a[@class='chart-row__artist']"
artist <- xpathSApply(xmlpage, searchfor, xmlValue)
length(artist)
```

```
## [1] 189
```

```
head(artist,6)
```

```
## [1] "\nThe Beatles\n"        "\nThe Rolling Stones\n"
## [3] "\nThe Monkees\n"        "\nThe Beatles\n"
## [5] "\nSoundtrack\n"         "\nSoundtrack\n"
```

The number of albums does not match the number of artists!

We must find out why...

Extracting data from Billboard.com

It turns out that some artists are listed within the `<span></span>` tag:

```
<span class="chart-row__artist">
Herb Alpert &amp;amp; The Tijuana Brass
</span>
```

But the class name is still the same `chart-row__artist`

We can easily deal with that...

Extracting data from Billboard.com

Here's the second attempt at extracting the artists:

```
searchfor <- "(//a|//span)[@class='chart-row__artist']"
artist <- xpathSApply(xmlpage, searchfor, xmlValue)
length(artist)
```

```
## [1] 200
```

```
head(artist,6)
```

```
## [1] "\nThe Beatles\n"                "\nThe Rolling Stones\n"
## [3] "\nThe Monkees\n"                "\nDiana Ross & The Supremes\n"
## [5] "\nThe Beatles\n"                "\nSoundtrack\n"
```

Works like a charm!

Extracting data from Billboard.com

We may also extract http links to the Artist's catalogue on Billboard

Note that the HTML `<a></a>` tag for the artist contains information other than the artist's name:

```
<a class="chart-row__artist" href="/music/the-beatles" data-tracklabel="Artist Name">
The Beatles
</a>
```

The `href` attribute of the `<a></a>` tag points to the Artist's catalogue

Extracting data from Billboard.com

Here's how we extract the links to artists catalogues:

```
searchfor <- "(//a|//span)[@class='chart-row__artist']"
artist_library <- xpathSApply(xmlpage, searchfor, xmlGetAttr, "href")
artist_library  <- paste0("https://www.billboard.com", artist_library )
artist_library [grep("NULL$", artist_library ) ] <- NA
head(artist_library ,6)
```

```
## [1] "https://www.billboard.com/music/the-beatles"
## [2] "https://www.billboard.com/music/the-rolling-stones"
## [3] "https://www.billboard.com/music/the-monkees"
## [4] NA
## [5] "https://www.billboard.com/music/the-beatles"
## [6] "https://www.billboard.com/music/soundtrack"
```

Extracting data from a webpage

- The ability to extract data from a webpage depends on how well structured the HTML code is
- First, inspect the HTML code within the web browser and locate examples of chucks of codes that contain the data of interest
- Then apply the search logic to extract the information of interest using the `XML` package, but be aware that things might not work from the first go
- You might need to modify the search criteria to deal with exceptions
- Always do some manual checks; Nothing replaces a human eye to validate the output

Chart data for entire 1968

Once we figure out how to extract data for one week, applying the same scheme over the entire year is straight forward:

It's important to write efficient, well organized, and easy to follow code!

```
chart_long <- data_frame(Album = character(), Artist = character(),
    Week = numeric(), Rank = numeric())
start_date <- as.Date("1968-01-06")
for (w in 1:52) {
    current_date <- start_date + 7 * (w - 1)
    url <- paste0("https://www.billboard.com/charts/billboard-200/",
        current_date)
    xmlpage <- htmlParse(rawToChar(GET(url)$content))
    album.title <- xpathSApply(xmlpage, "//h2[@class='chart-row__song']",
        xmlValue)
    album.author <- xpathSApply(xmlpage, "(//a|//span)[@class='chart-row__artist']",
        xmlValue)
    chart_long <- chart_long %>% bind_rows(data_frame(Album = album.title,
        Artist = album.author, Week = w, Rank = 1:200))
    print(paste0("chart for week ", w, "fetched"))
    flush.console()
}
```

Is web scraping legal?

- The legality of web scraping is an intricate issue
- On the one hand, search engines such as Google and Bing rely on web scraping to catalogue the web
- At the same time, many websites terms of service prohibit unauthorized scraping
- Note that while scraping might not be illegal, the data itself could be copyrighted. Do not be aggressive in the extent of the data you retrieve from a website
- A poorly written scrapping script could bring down a web site by overwhelming the server with requests. Avoid `while` loops and always test the script on a small subset of the data before running it on the full set

Now is your turn to practice!

The president of the united states is an avid tweeter. We'd like to fetch all the tweets from his twitter webpage to do some textual analysis on. For this first task, all you need to do is:

Retrieve the twitter page of @realDonaldTrump

Extract from the page all tweets written by @realDonaldTrump

All the President's tweets

Retrieving the twitter feed for @realDonaldTrump

```
url <- "https://twitter.com/realDonaldTrump?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor"
xmlpage <- htmlParse(rawToChar(GET(url)$content))
length(capture.output(xmlpage))
```

```
## [1] 6934
```

```
capture.output(xmlpage)[1:5]
```

```
## [1] "<!DOCTYPE html>"
## [2] "<html lang=\"en\" data-scribe-reduced-action-queue=\"true\">"
## [3] "<head>"
## [4] "<meta charset=\"utf-8\">"
## [5] "<script nonce=\"7vWGUMRlGErq4AVBM38Z/Q==\">"
```

All the President's tweets

Next, we extract the tweets:

```
searchfor <- "//p[@class='TweetTextSize TweetTextSize--normal js-tweet-text tweet-text']"
tweets <- xpathSApply(xmlpage, searchfor, xmlValue)
length(tweets)
```

```
## [1] 20
```

```
head(tweets, 5)
```

```
## [1] "Just arrived @NASKeyWest! Heading to a briefing with the Joint Interagency Task Force South, NOR
## [2] "Governor Jerry Brown announced he will deploy <U+0093>up to 400 National Guard Troops<U+0094> to
## [3] "Thank you San Diego County for defending the rule of law and supporting our lawsuit against Cali
## [4] "Great meeting with Prime Minister Abe of Japan, who has just left Florida. Talked in depth about
## [5] "It was my great honor to host my friend @JPN_PMO @AbeShinzo and his delegation at Mar-a-Lago for
```

But some of these might be retweets of other users

All the President's tweets

Here's a more proper approach:

```
library(dplyr)
searchfor <- "//p[@class='TweetTextSize TweetTextSize--normal js-tweet-text tweet-text']"
tweets <- xpathSApply(xmlpage, searchfor, xmlValue)
searchfor <- "//strong[@class='fullname show-popup-with-id u-textTruncate ']"
tweets_by <- xpathSApply(xmlpage, searchfor , xmlValue)
df_tweets <- data_frame(Tweet =1:length(tweets), Text=tweets, Author=tweets_by)
df_tweets$Author
```

```
##  [1] "Donald J. Trump"     "Donald J. Trump"     "Donald J. Trump"
##  [4] "Donald J. Trump"     "Donald J. Trump"     "Department of State"
##  [7] "Donald J. Trump"     "Donald J. Trump"     "Donald J. Trump"
## [10] "Donald J. Trump"     "Donald J. Trump"     "Donald J. Trump"
## [13] "Donald J. Trump"     "<U+5B89><U+500D><U+664B><U+4E09>" "Ivanka Trump"
## [16] "Donald J. Trump"     "Donald J. Trump"     "Donald J. Trump"
## [19] "Donald J. Trump"     "Donald J. Trump"
```

All the President's tweets

```
df_tweets %>% filter(Author == "Donald J. Trump") %>% select(Text)
```

```
## # A tibble: 17 x 1
##                                                                      Text
##                                                                     <chr>
##  1 Just arrived @NASKeyWest! Heading to a briefing with the Joint Interagency
```

```
##  2 Governor Jerry Brown announced he will deploy <U+0093>up to 400 National Guard Tro
##  3 Thank you San Diego County for defending the rule of law and supporting our
##  4 Great meeting with Prime Minister Abe of Japan, who has just left Florida.
##  5 It was my great honor to host my friend @JPN_PMO @AbeShinzo and his delegat
##  6 Great working luncheon with U.S. and Japanese Delegations this afternoon!pi
##  7 Prime Minister @AbeShinzo of Japan and myself this morning building an even
##  8 Best wishes to Prime Minister @Netanyahu and all of the people of Israel on
##  9 Slippery James Comey, the worst FBI Director in history, was not fired beca
## 10 Mike Pompeo met with Kim Jong Un in North Korea last week. Meeting went ver
## 11 A sketch years later about a nonexistent man. A total con job, playing the
## 12 There is a Revolution going on in California. Soooo many Sanctuary areas wa
## 13 States and Cities throughout our Country are being cheated and treated so b
## 14 Today<U+0092>s Court decision means that Congress must close loopholes that block
## 15 ....Congress <U+0096> House and Senate must quickly pass a legislative fix to ensu
## 16 While Japan and South Korea would like us to go back into TPP, I don<U+0092>t like
## 17 Pastor Andrew Brunson, a fine gentleman and Christian leader in the United
```

Flash-forward

In Lesson 9, we'll learn how to extract predictive value from text

Figure 2:

Text processing

- Even though we were able to capture data from a webpage into an R data frame, our job is not done
- Textual data is messy by default; we might need to do some cleaning to make it more presentable
- Example, in the Billboard chart data, album titles contain the newline character `\n`

```
## [1] "\nThe Beatles\n"
```

- And in the twitter data, some tweets had web addresses included within the text
- We are not able yet to do full treatment of textual data, but we can do some gentle processing with Base R and `stringr`

The gsub function

Use `gsub` to match and substitute a pattern within a string:

```
txt = "Hello world! Hello world! Hello world!"
gsub("Hello", "Goodbye", txt)
```

```
## [1] "Goodbye world! Goodbye world! Goodbye world!"
```

Note that the pattern specified can be a **regular expression**:

```
txt = "a man and a woman"
gsub("man", "woman", txt)
```

```
## [1] "a woman and a wowoman"
```

```
gsub("\\bman\\b", "woman", txt)
```

```
## [1] "a woman and a woman"
```

The gsub function

With `gsub` we can clean the artist names in the Billboard data:

```
head(artist, 6)
```

```
## [1] "\nThe Beatles\n"                "\nThe Rolling Stones\n"
## [3] "\nThe Monkees\n"                "\nDiana Ross & The Supremes\n"
## [5] "\nThe Beatles\n"                "\nSoundtrack\n"
```

```
artist <- gsub("\\n", "", artist)
head(artist, 6)
```

```
## [1] "The Beatles"                "The Rolling Stones"
## [3] "The Monkees"                "Diana Ross & The Supremes"
## [5] "The Beatles"                "Soundtrack"
```

Now is your turn to practice!

Starting from the full set of tweets extracted from @realDonaldTrump twitter page, use the `gsub` function to remove all occurances of the definite article "the" from the tweets.

All the President's tweets

Here's a possible solution to the exercise:

```
gsub("[ ]{2,}", " ", gsub("\\bthe\\b", "", df_tweets$Text, ignore.case = TRUE))
```

```
##  [1] "Just arrived @NASKeyWest! Heading to a briefing with Joint Interagency Task Force South, NORTH
##  [2] "Governor Jerry Brown announced he will deploy <U+0093>up to 400 National Guard Troops<U+0094> 
##  [3] "Thank you San Diego County for defending rule of law and supporting our lawsuit against Califo
##  [4] "Great meeting with Prime Minister Abe of Japan, who has just left Florida. Talked in depth abo
##  [5] "It was my great honor to host my friend @JPN_PMO @AbeShinzo and his delegation at Mar-a-Lago f
##  [6] ".@POTUS Trump thanks Prime Minister @AbeShinzo for his support, discusses U.S.-Japan cooperati
##  [7] "Great working luncheon with U.S. and Japanese Delegations this afternoon!pic.twitter.com/ywU2C
##  [8] "Prime Minister @AbeShinzo of Japan and myself this morning building an even deeper and better 
##  [9] "Best wishes to Prime Minister @Netanyahu and all of people of Israel on 70th Anniversary of yo
## [10] "Slippery James Comey, worst FBI Director in history, was not fired because of phony Russia inv
## [11] "Mike Pompeo met with Kim Jong Un in North Korea last week. Meeting went very smoothly and a go
## [12] "A sketch years later about a nonexistent man. A total con job, playing Fake News Media for Foo
## [13] "There is a Revolution going on in California. Soooo many Sanctuary areas want OUT of this ridi
## [14] "<U+30D5><U+30ED><U+30EA><U+30C0><U+306B><U+5230><U+7740><U+3057><U+3001><U+65E9><U+901F><U+30C8
## [15] "This year<U+0092>s #TaxDay is last time you<U+0092>ll have to file your taxes through an outda
## [16] "States and Cities throughout our Country are being cheated and treated so badly by online reta
## [17] "Today<U+0092>s Court decision means that Congress must close loopholes that block removal of d
## [18] "....Congress <U+0096> House and Senate must quickly pass a legislative fix to ensure violent c
## [19] "While Japan and South Korea would like us to go back into TPP, I don<U+0092>t like deal for Un
## [20] "Pastor Andrew Brunson, a fine gentleman and Christian leader in United States, is on trial and
```

The grep function

grep is similar to gsub in the syntax of the search pattern, but it is used solely for searching (no string substitution is done)

How many of the top albums in 1968 are Beatles albums?

```
result <- grep("beatles", artist, ignore.case = TRUE)
result
```

```
## [1]   1   5 153
```

```
album[result]
```

```
## [1] "Magical Mystery Tour (Soundtrack)"
## [2] "Sgt. Pepper's Lonely Hearts Club Band"
## [3] "Revolver"
```

The grepl function

grepl is equivalent to grep but returns a Boolean vector instead of indices of matched elements:

```
result <- grep("beatles", artist, ignore.case = TRUE)
result
```

```
## [1]   1   5 153
```

```
result <- grepl("beatles", artist, ignore.case = TRUE)
result[1:10]
```

```
##  [1]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
```

Now is your turn to practice!

Starting from the full set of tweets extracted from @realDonaldTrump twitter page, use the `grep` or `grepl` function to find tweets where Trump mentions himself in the tweet.

All the President's tweets

Here's a possible solution to the exercise:

```
grep("\\bTrump\\b", df_tweets$Text, ignore.case = TRUE, value = TRUE)
```

```
## [1] ".@POTUS Trump thanks Prime Minister @AbeShinzo for his support, discusses U.S.-Japan cooperatio
## [2] "Prime Minister @AbeShinzo of Japan and myself this morning building an even deeper and better r
```

Regular expressions

- A regular expression is special notation for a search pattern
- The regular expression notation is universal (i.e. it is independent of the programming language)
- Regular expressions are characterized by their compactness and efficiency
- Examples of regular expression notation: `\b` represents word boundary, `^` beginning of a string, `$` end of a string, `[a-z]` any character from a to z (lower case)
- These are just few examples; the complete set of notation is very expansive and is best learned on a needs basis

Regular expressions

Albums that have a number in the title:

```
grep("[0-9]", album, ignore.case = TRUE, value = TRUE)
```

```
## [1] "Bee Gees' 1st"                  "Album 1700"
## [3] "Sergio Mendes & Brasil '66"     "$1,000,000.00 Weekend"
## [5] "Evergreen, Vol. 2"              "16 Original Big Hits, Volume 7"
## [7] "Best Of The Beach Boys, Vol. 2" "16 Original Big Hits, Volume 8"
```

Albums that have a one-word title:

```
grep("^[a-z0-9]+$", album, ignore.case = TRUE, value = TRUE)
```

```
##  [1] "Camelot"       "Headquarters"  "Revenge"       "SRO"
##  [5] "Clambake"      "Respect"       "Claudine"      "United"
##  [9] "Wonderfulness" "Equinox"       "Flowers"       "Joan"
## [13] "Wildflowers"   "Revolver"      "Collage"       "Collections"
## [17] "Camelot"
```

Regular expressions

Albums that have a repeated word:

```
grep("\\b([a-z0-9]+) \\1", album, ignore.case = TRUE, value = TRUE)
```

```
## [1] "Pata Pata"
```

Bands whose name is a single word preceded by "the":

```
grep("^the [a-z0-9]+$", artist, ignore.case = TRUE, value = TRUE) %>% unique()
```

```
##  [1] "The Beatles"     "The Monkees"     "The Doors"
##  [4] "The Turtles"     "The Temptations" "The Cowsills"
##  [7] "The Rascals"     "The Byrds"       "The Association"
## [10] "The Lettermen"   "The Animals"     "The Ventures"
## [13] "The Hollies"     "The Who"         "The Yardbirds"
```

Now is your turn to practice!

Starting from the full set of tweets extracted from @realDonaldTrump twitter page, use the string search functions introduced in this lesson to display the tweets that end with an exclamation mark.

All the President's tweets

Here's a possible solution to the exercise:

```
grep("!$", df_tweets$Text, value = TRUE)
```

```
## [1] "Governor Jerry Brown announced he will deploy <U+0093>up to 400 National Guard Troops<U+0094> t
## [2] "Great meeting with Prime Minister Abe of Japan, who has just left Florida. Talked in depth abou
## [3] "Best wishes to Prime Minister @Netanyahu and all of the people of Israel on the 70th Anniversar
## [4] "Slippery James Comey, the worst FBI Director in history, was not fired because of the phony Rus
## [5] "Mike Pompeo met with Kim Jong Un in North Korea last week. Meeting went very smoothly and a goo
## [6] "There is a Revolution going on in California. Soooo many Sanctuary areas want OUT of this ridic
## [7] "States and Cities throughout our Country are being cheated and treated so badly by online retai
## [8] "....Congress <U+0096> House and Senate must quickly pass a legislative fix to ensure violent cr
## [9] "Pastor Andrew Brunson, a fine gentleman and Christian leader in the United States, is on trial
```

Now is your turn to practice!

Starting from the full set of tweets extracted from @realDonaldTrump twitter page, use the string search functions introduced in this lesson to display the tweets that contain acronyms (e.g. USA, E.U, NRA, etc.).

All the President's tweets

Here's a possible solution to the exercise:

```
grep("([A-Z][.]?){2,}", df_tweets$Text, value = TRUE)
```

```
## [1] "Just arrived @NASKeyWest! Heading to a briefing with the Joint Interagency Task Force South, NO
## [2] "It was my great honor to host my friend @JPN_PMO @AbeShinzo and his delegation at Mar-a-Lago fo
## [3] ".@POTUS Trump thanks Prime Minister @AbeShinzo for his support, discusses U.S.-Japan cooperatio
## [4] "Great working luncheon with U.S. and Japanese Delegations this afternoon!pic.twitter.com/ywU2CE
```

```
## [5] "Slippery James Comey, the worst FBI Director in history, was not fired because of the phony Russ
## [6] "There is a Revolution going on in California. Soooo many Sanctuary areas want OUT of this ridicu
## [7] "<U+30D5><U+30ED><U+30EA><U+30C0><U+306B><U+5230><U+7740><U+3057><U+3001><U+65E9><U+901F><U+30C8
## [8] "This year<U+0092>s #TaxDay is the last time you<U+0092>ll have to file your taxes through an ou
## [9] "While Japan and South Korea would like us to go back into TPP, I don<U+0092>t like the deal for
```

```
unlist(str_extract_all(df_tweets$Text, "([A-Z][.]?){2,}")) %>% head(20)
```

```
##  [1] "NASK"      "NORTHCOM"  "SOUTHCOM." "IX"        "JPN"
##  [6] "PMO"       "POTUS"     "U.S."      "ONHF"      "KC"
## [11] "U.S."      "CE"        "FBI"       "NO"        "COLLUSION"
## [16] "OUT"       "NOW"       "ARS"       "BYE"       "BYE"
```

The `str_extract` function

The `str_extract_all` function of the `stringr` package is used to extract matched patterns from text

`str_extract` uses the same regular expression notation we're now familiar with

```
txt <- c("this sentence has some words",
    "some words are short and some are long",
    "one, two, three, red, blue, green")
str_extract_all(txt, "\\b[a-zA-z]{3,3}\\b") %>%
    unlist()
```

```
## [1] "has" "are" "and" "are" "one" "two" "red"
```

Now is your turn to practice!

Starting from the full set of tweets extracted from @realDonaldTrump twitter page, use the string search functions introduced in this lesson to extract phrases with title caps.

All the President's tweets

Here's a possible solution to the exercise:

```
str_extract(df_tweets$Text, "(\\b[A-Z][a-z]+\\b[ ]?){3,}") %>%
    unlist() %>% na.omit() %>% head()
```

```
## [1] "Joint Interagency Task Force South"
## [2] "Governor Jerry Brown "
## [3] "San Diego County "
## [4] "Prime Minister Abe "
## [5] "Trump International Golf Club"
## [6] "Slippery James Comey"
```

Note that the search pattern assumes that a web address starts with http or https

Now is your turn to practice!

Starting from the full set of tweets extracted from @realDonaldTrump twitter page, use the string search functions introduced in this lesson to extract web addresses embedded in the tweets.

All the President's tweets

Here's a possible solution to the exercise:

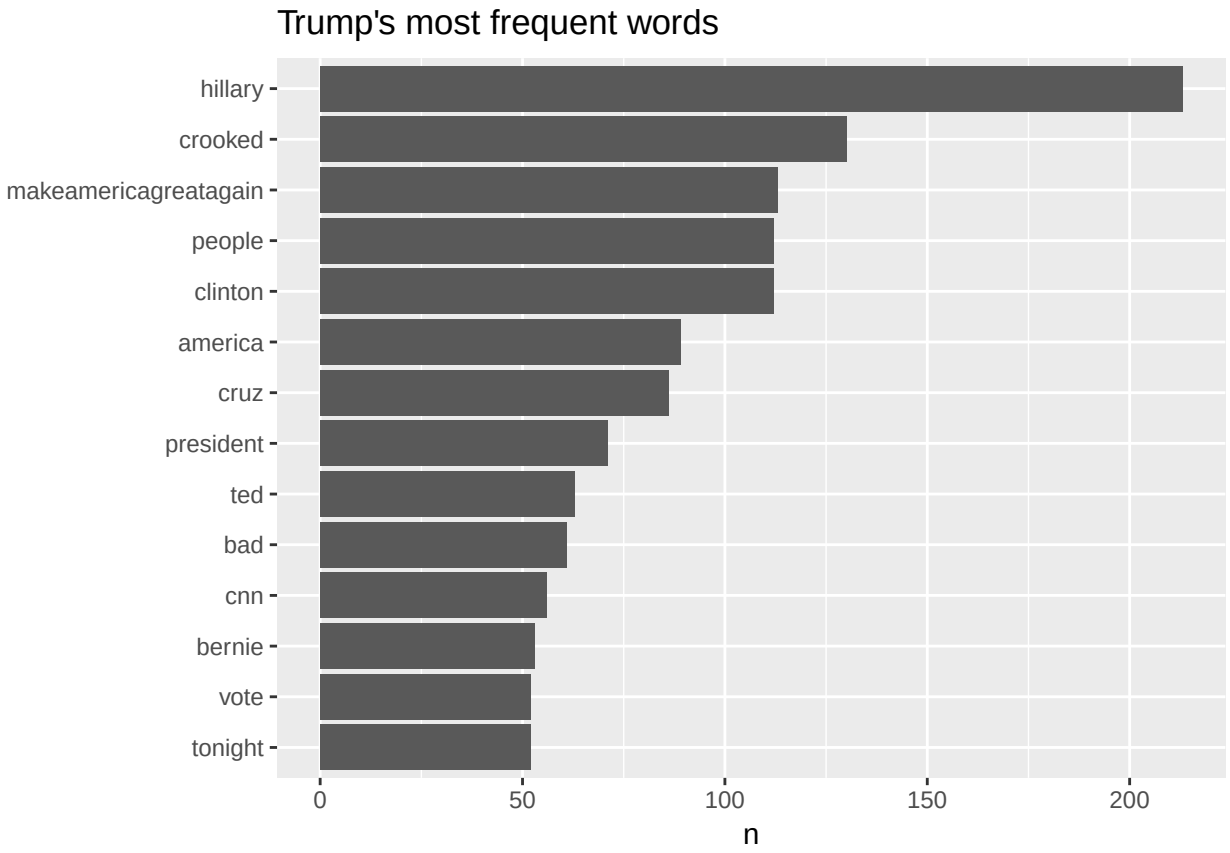
```
str_extract(df_tweets$Text, "\\bhttp(s)?:[a-zA-Z0-9:/.]+\\b") %>%
    unlist() %>% na.omit() %>% head()
```

```
## [1] "http://WhiteHouse.gov"
## [2] "https://twitter.com/shennafoxmusic/status/986544764395900928"
```

Note that the search pattern assumes that a web address starts with http or https

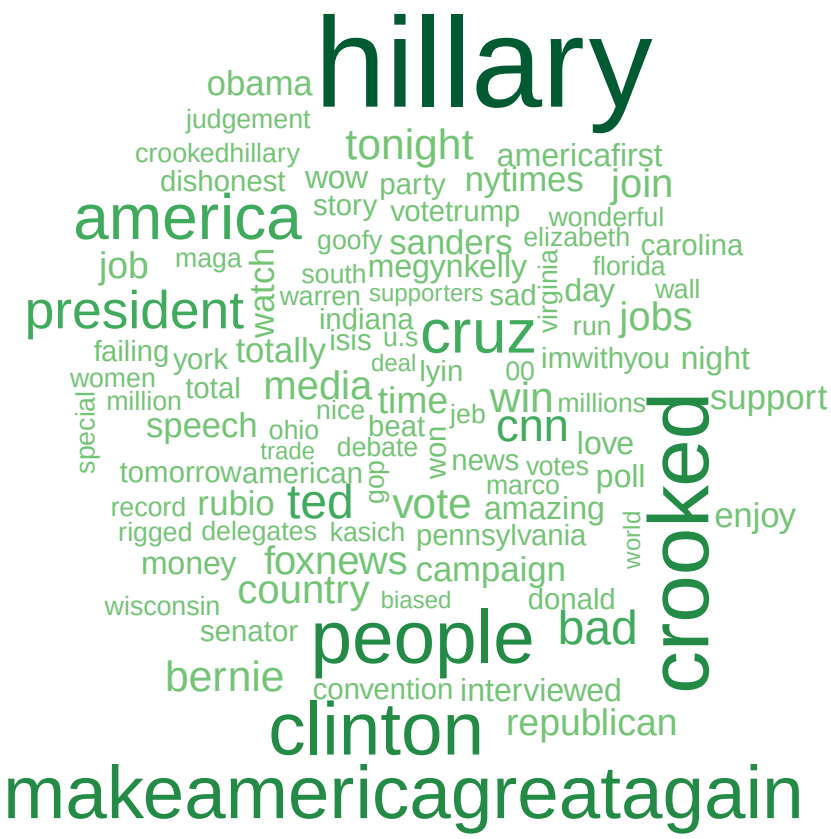
Flash-forward

In Lesson 9, we'll learn how to model text using **bag of words** approach



Flash-forward

In Lesson 9, we'll learn how to build **word clouds**



Concluding remarks

- By now, we should be able to obtain data that exists in the form of a well structured csv file as well as data on a webpage
- We learned how the `XML` package simplifies the job of extracting data from the web
- Once the data is loaded in R as data frame, we are able to manipulate it using functions from `dplyr` and `tidyr`
- Textual data, is more messy than numeric or categorical data
- But we are able to do some basic text manipulation with Base R and `stringr`
- We were introduced to regular expressions as a way to efficiently search for patterns in text